

# GWT From Scratch – Day 1

## Getting To 'Hello'

The aim of today's session is to get you started with GWT.

### **You Will**

- Download Eclipse
- Set up a structure for your GWT work area
- Get Eclipse up and running
- Download GWT
- Download a ready-made Project
- Import it into Eclipse
- Configure Eclipse
- Understand the parts of a project
- Run the ready-made project
- Learn how to clone a basic project so you don't have to start from scratch every time

### **Questions?**

My ideal is to have no questions at all because everything is perfectly clear, but if you have any questions, then please contact me. I would like to consider the course 'complete' within the limits I have set for it. In other words, I am here to supplement the course if, in any way, it is not well enough explained to get everyone through it. I intend to use the feedback to improve the course and reduce the questions. So any questions are very welcome.

### **Any Problems**

[rx01-day1@examples.roughian.com](mailto:rx01-day1@examples.roughian.com)

© Ian Bambury 2008

## The "It's Not My Fault" Stuff

### ALL RIGHTS RESERVED.

I'd be grateful if you didn't sell or give this document away or post any of its content on a web site. I have a box of code-weevils and I will set them on any site that does. But an extract as an illustrative quote in the context of a review plus a link to my site, that's OK.

### DISCLAIMER AND/OR LEGAL NOTICES:

The information presented herein represents the view of the author as of the date of publication, whenever that was. While every attempt has been made to verify the information provided here, the author does not assume any responsibility for errors, inaccuracies, omissions or outright blatant lies contained herein. Any slights of people or organizations are completely intentional. If I say they are a bunch of bastards, then that is what I think they are, and you can quote me. If advice concerning legal or related matters is needed, the services of a fully qualified professional should be sought. I also double as an unqualified psychiatrist if things get really bad. Any reference to any person or business whether living or dead is purely coincidental. Except when it's not.

## **IDE - Eclipse**

You can write GWT applications in Notepad or vi if you want, but you will have a much easier time of it if you get yourself an IDE (Integrated Development Environment).

You are, of course, free to use anything or nothing, but this course assumes that you will have Eclipse. I make that assumption because I have to choose one, Eclipse is the most popular, it's the one I use, it's available on all platforms that support Java (which is pretty much everything), it's pretty darn good even if it does lock up for a while sometimes, and it's free.

### **Download Eclipse**

<http://www.eclipse.org/downloads/>.

I use Eclipse Classic 3.3.2 but Eclipse IDE for Java Developers would also be suitable. Simply unzip it somewhere and there you go.

## **Java Runtime Environment**

You will need a Java environment. You can get the latest version from

### **Download JRE**

<http://www.java.com/en/download/index.jsp>

It's a simple install. To run GWT 1.5, you will need Java 5 or above, the above link is OK for that.

## **File Structure**

Now I work a little 'unclassically' in that I have GWT files (and others) in each of the GWT project directories. You don't need to do this, you can just have one lot. The advantage of doing it my way is that you can use relative paths and \*that\* means that it is portable - you can move the whole lot to another disk on another machine in a different directory, and it will still work.

So I have a GWT directory where absolutely everything to do with GWT goes:

[/gwt/](#)

and I have Eclipse in

[/gwt/eclipse/](#)

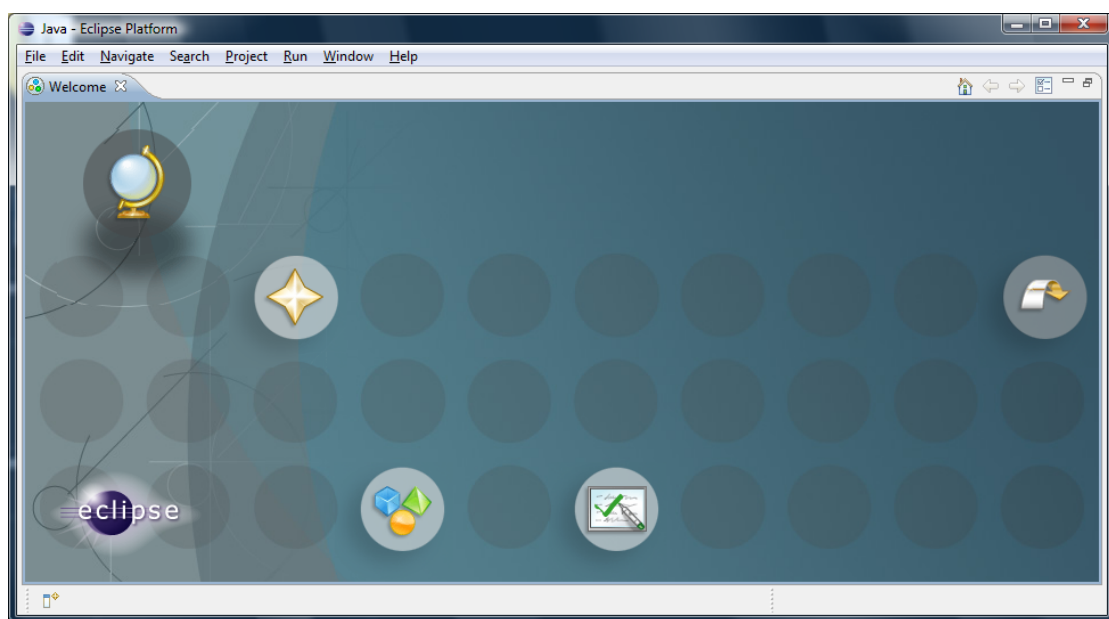
but Java is just in the Program Files directory

If you don't set up this way, then you'll have to translate paths, but it won't be difficult, and there aren't many times you'll need to do it at all.

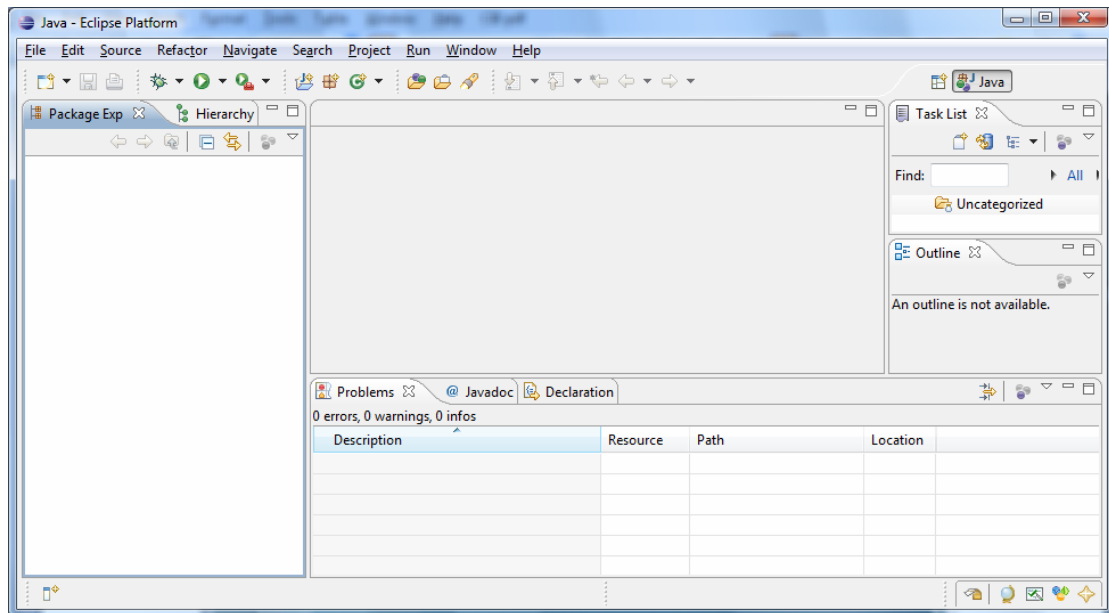
When you get to needing a web-server and database, it makes things a lot easier but you take a hit on your backups time. Talking of which, have you tried East-Tech 2007? It's free and very adaptable. Another good one is SyncBack, also free. But I digress...

## **Starting Eclipse**

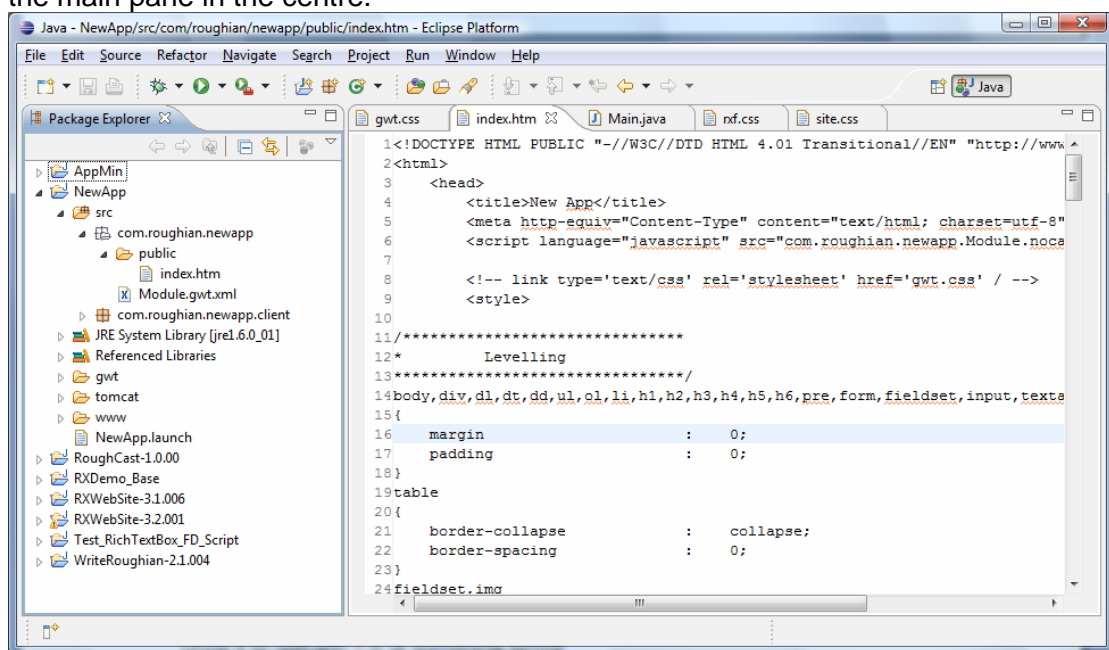
Double-click `/gwt/eclipse/eclipse.exe` to fire it up. It takes a moment but then comes up with a Welcome screen that you'll probably never see again.



You need to click on the Workbench icon (a kind of bouncy arrow thing over on the right-hand side)



If you are like me, you will get rid of all the panels except the Package Explorer and the main pane in the centre.



The thing is pretty straightforward if you cut it down to the basics like I have. You can always get the other options back if you want them.

On the left you have the Package Explorer which allows you to select a package or the files inside it, and on the right you have an editor with tabs for the individual files at the top.

Of course, Eclipse is much more sophisticated than that but for our purposes, this is all you will need.

As the course goes on, I will explain how to use the various menu options as we run into them. But for now all you need to do is make sure that Eclipse is working.

## GWT

At the time of writing, the latest stable version is 1.4, but as 1.5 milestone 2 is available, we'll use that. It will become 'proper' soon and 1.4 will be outdated, and this \*is\* only a course so it doesn't need to be production standard.

### Download GWT

<http://code.google.com/p/google-web-toolkit/downloads/list?can=4&q=version:1.5>

Unzip it to `/gwt/gwt-1.5/` or something similar

The Google Web Toolkit allows you to write applications as if you were writing Java.

The syntax is identical, but because everything gets translated into JavaScript eventually, and because Google haven't had time to emulate everything that is available in Java, there are quite a lot of things that you can do in Java that won't work in GWT.

For example, you can't use multithreading. In Java you can because Java allows multithreading. In GWT you can't use multithreading because JavaScript runs as a single thread and therefore GWT has no way to translate Java's multithreading model into JavaScript. It is just impossible.

If you come from a Java background, then you might find that GWT is a little bit frustrating because not everything is there. If you come from a Web background where you have been using JavaScript or something else, then you probably won't because you will only be trying to do what you have previously been doing in JavaScript (or whatever).

For this reason, you're probably better off not knowing Java! Provided that you have used a curly-bracketed language, then you won't have any problem with GWT, and you will probably just be pleasantly surprised (or maybe even amazed) with what you can do with GWT.

If you're never used Java or Eclipse before, but have a reasonable background in programming, then don't worry. I myself had never used either before I started, and I didn't find it a problem at all.

Hi Ian.

The only issues I had was that I am running OS X 10.5 (Leopard) and GWT requires a specific version to run under leopard.

The other issue I had was that on OS X gwt requires an additional launch param. I had to add:

```
<stringAttribute key="org.eclipse.jdt.launching.VM_ARGUMENTS" value="-XstartOnFirstThread"/>
```

to the .launch file

Matt

# The First Project

Now you can, if you want, create a project the Google way by running the applicationCreator and projectCreator commands, but they are such a pain in the arse that I am not going to cover them.

For one thing, to make the commands easier, they use the project name for everything, the result being that you can't see how things fit together - is MyProject the module name minus the '.xml'? Or something else minus something else?

For another thing, once you have set up a basic project, you will never use those commands again - you will just copy the basic project - so why do you have to go through all that hassle when I have already been through it and can just give you a working project with all the names changed to make sense?

You don't, of course.

## **Download The Project**

You can just download it from here.

<http://examples.roughian.com/rx01/AppMin.zip>

Once you have it, then unzip it to

[/gwt/workspace/](#)

where you should have a new directory called

[/gwt/workspace/AppMin/](#)

## **Alternative Method If You Can't Download zips**

You need the files listed below in these exact locations - capitalisation must be exact, even in Windows. Each item in the list is followed by a link to the correct file on the Roughian Examples server.

Note that the index.htm file will open and appear blank, but if you save it, (or view source) you will see it has real content. The others will probably display in your browser (they are all XML, more or less), so save these or view the source and cut and paste it.

If you still can't get the files, let me know and we'll try to work something out.

```
AppMin\src\com\roughian\appmin\Module.gwt.xml
```

<http://examples.roughian.com/rx01/Module.gwt.xml>

```
AppMin\src\com\roughian\appmin\client>Main.java
```

<http://examples.roughian.com/rx01/Main.java>

```
AppMin\src\com\roughian\appmin\public\index.htm
```

<http://examples.roughian.com/rx01/index.htm>

```
AppMin\.classpath
```

<http://examples.roughian.com/rx01/.classpath>

```
AppMin\.project
```

<http://examples.roughian.com/rx01/.project>

```
AppMin\AppMin.launch
```

<http://examples.roughian.com/rx01/AppMin.launch>

**(End of Alternative)**



## **Import It Into Eclipse**

To import the project into Eclipse, right-click an empty area of the Package Explorer and choose Import...

Alternatively, you can use File | Import...

You will get to the Import dialog box. Double-click 'Existing Projects into Workspace'. Use the Browse button at the top to navigate to

[/gwt/workspace/AppMin/](#)

...so that AppMin is highlighted. Then click OK. Then click Finish.

You will then have imported the application, and you should be to see it in the Package Explorer.

## **The Public Directory**

Double click on the package name (or just expand it), then double-click on 'src', 'com.roughian.appmin', and 'public'.

The public directory contains everything that you want to go onto the web server. So here then, you will find our only HTML page - index.htm

## **File Associations**

By default, for some reason, Eclipse will show this rendered as a webpage. We need to tell it that we want to see it as text that we can edit.

In the main menu, choose Window | Preferences. In the Preferences dialog box, open up General | Editors, and then choose File Associations.

Click the Add button and adds a new file type of \*.htm.

Highlight it in the File types box, and click the Add button next to the Associated editors box. Scroll down to Text Editor, highlighted, and click OK.

Do the same thing for \*.html files if that is your preferred extension, and while you're at it do the same thing for \*.launch files which will going to need in a minute.

## The Index File Explained

A bog-standard GWT application has just the one HTML file. The entries are as follows:

- a doctype
- a title
- a meta-tag to ensure we have utf-8 character set
- a script tag which fires up everything that GWT produces
- a commented-out link for a CSS sheet
- an empty pair of style tags
- an iframe which is used if you want to include history support (this allows the use of the browser's forward and back buttons within your application)
- and a div with the ID of 'slot'.

The style tags are there just so that you can add a bit of style if you're playing around with a widget.

The div with the ID of 'slot' is referred to in the Java file which we will look at next.

## The Java File Explained

Double click on 'com.roughian.appmin.client'. This directory contains all the files needed for GWT to create your application. There can also be a 'server' directory, but this is for pure Java which is run on the server and is not used by GWT when it is producing JavaScript.

Double click on Main.java and you will see what goes into that.

The package name, this matches up with the package name in the Package Explorer and also with the directory structure.

- Three imports.
- A class declaration of 'Main' - this has to match up with the filename.
- A method called onModuleLoad().
- And a single line of code.

If you're not familiar with Java, then what you need to know is that the package name is just a name for this 'chunk' of code, and that imports are the way that your chunk of code tells the system that it wants to use other chunks of code.

The Main class implements EntryPoint. What that means is that when your application runs, GWT will ensure that the onModuleLoad() method will be run as soon as the module has loaded. We could have more than one class in a project that implements EntryPoint, but you don't need to worry about that at the moment.

## The One Line Of Code

This one line of code will seem really elementary to you later on, but there are a lot of things in it that you will probably need explained right now.

## The RootPanel

RootPanel is GWT's way of letting you get hold of the body tag in your HTML.

The 'get' part is a method which will look for elements within your HTML which have the ID that you pass to it. If you don't pass on ID to it, then it will return the whole of the body tag contents.

So in this example, it will look for something with the ID of 'slot', and if you remember back to looking at the HTML, there was a div with that ID, and so that is what you would get.

WARNING: there is a problem with RootPanel - probably not anything that you have to worry about right now, but RootPanel will cache anything that you 'get' from it. This means that if you 'get' a parent containing a child with the ID of, say, 'slot', then you 'get' that child element, then you change the parent element and add another one which also contains a child called 'slot' and then you try to 'get' the new element called 'slot', then you will get the cached (i.e. the old) version of 'slot'. Like I say, don't worry about it now, but let it sit in the back of your mind so that if you hit a problem later on, it might come back to you.

Right, back to our one line of code. So we've been to the RootPanel and we've got slot called 'slot', and we are going to add something to it. In this particular case, we are going to add a label with the text "Hello".

If you're not familiar with Java, then the 'new' keyword tells the compiler to create a new object, in this case a label. The 'Label()' part tells the compiler what kind of object to create, and the bit between the parentheses allows you to pass parameters to the bits of the Label code which is dedicated to creating a new Label. In the case of said Label, if you pass it a String, then it will shove the text you pass to it into the label.

## Actually Running The Thing

Here we are 2/3rds of the way through, and we haven't actually run this one line of code yet.

There is just a little bit more we have to do before we can actually run the program.

## If You Want A Flexible Set-Up

If you want a flexible set up like mine, then you need to have the GWT files within the project directory. So go to the place where you unzipped the GWT files and select the following files:

- gwt-dev-windows.jar
- gwt-ll.dll
- gwt-user.jar
- swt-win32-3235.dll

and paste them into the `/gwt/workspace/AppMin/gwt/` directory.

When you want to use a different version of GWT, just go and get these the same files from wherever you unpacked that version, and copy them to the project's /gwt/ directory.

### **If You Want An Ordinary Set-Up**

With an ordinary kind of setup, you have the GWT files in one place and every project uses the same files. If you don't move your files around, then this is probably the most sensible option, but if you want to be able to work on projects on different computers which are set up with different directory structures, or you want to go to take your files and work on them on a flash drive, or if you think you might ever change your directory names, then go for the previous option.

For an ordinary setup, then:

With some part of your project selected in the Package Explorer, choose Project | Properties from the menu system. In the dialog box that appears, select Java Build Path. Under the Libraries tab, select each of the GWT jar files in turn, click Edit... and point it at the appropriate file where you unpacked GWT.

When you want to use a different version of GWT, just go and do the same as you have done here, but point these file names to the new version.

### **A Note For Linux Users**

This in from John who edited the .classpath file directly in order to get the classpath entries right on Linux:

Hi Ian.

For Linux, if I take your AppMin just as it comes, and change .classpath

a) change the word "windows" to "linux"

b) set the \_absolute\_ path to the two jar files

everything works like a charm.

```
<classpathentry kind="src" path="src"/>
<classpathentry kind="con" path="org.eclipse.jdt.launching.JRE_CONTAINER"/>
<classpathentry kind="lib" path="/xxxxx/gwt-user.jar"/>
<classpathentry kind="lib" path="/xxxxx/gwt-dev-linux.jar"/>
<classpathentry kind="output" path="bin"/>
```

Thanks.

John

### **Run It (Finally!)**

The easiest way to run it if you have a recent version of Eclipse, is:

- make sure some part of the project is selected in the Package Explorer
- click on the Run button

The buttons are in the order New, Save, Print, Debug, Run - it looks like a media player 'play' button.

What should happen is that two windows will, after a while, appear. The first one is the Shell window, and this is where messages appear when things go wrong, and occasionally when things go right. In this case you should get a message saying that HTTP has been started on port 8888.

The second window is the hosted browser and will show you your application. In this particular case, you should get the message 'Hello'.

## **Copying A Project**

As I said before, creating a new project by using the creation tools supplied by Google is a right pain, and doesn't produce a very well-named project.

What I tend to do, is simply to copy the project in Eclipse, and change a couple of names in a couple of files.

It may look quite complicated written out as it is below, but you'll soon get the hang of it, and really all it comes down to is pressing F2 a couple of times to rename a couple of things and then updating a few references. If you want to try it with the creators that Google supplies, then give it a go, and see you back here when you get fed up.

### **Copy The Project As Something Else**

With the actual project selected in the Project Explorer (i.e. AppMin) press Control+C, Control+V.

Change the name to whatever you want the new project to be called and click OK. I'm assuming you called it NewApp for the purposes of the next few paragraphs, but substitute your own new name, of course.

### **Change The References**

Open up the NewApp project and select the last file in that directory which is called AppMin.launch. Press F2 and rename it NewApp.launch. You can actually call this file anything you like, but it makes sense to call it the same as the project.

Double-click the newly named file and find an occurrence of the string 'AppMin' (note the capitalisation). The easiest place to find it is at the end of the second to last line.

Press Control+F to bring up the Find/Replace dialogue. The Find text box should be filled in for you, so fill in the Replace With text box with the new project name, and click Replace All. There should be four replaced occurrences.

### **Change The Package Name**

You don't actually have to do this step if you are just playing around, although there is a slight possibility that if you don't do it, then when you try to run the project you'll get some previous project with the same package name, but I haven't come across this very often.

Open up the /src/ directory in the new project and select the first entry ('com.roughian.appmin'). Press F2 to rename it. For the purposes of these instructions I'm assuming that you're going to change it to 'com.roughian.newapp'.

In the Rename Package dialogue, and to the new name in the text box, and ensure that you have checked Update references, and Rename subpackages. Click OK.

## Update Package Name References

There are three places where you have to update the package name manually. The rest have been done automatically by Eclipse.

- In the index.htm file you have to modify the script tag.
- In the Module.gwt.xml file you have to change the entry point tag.
- In the .launch file you have to change the third-to-last line.

And you are done.

## **That's All For Today**

This is quite convenient place to leave it. By now, if you've read it through once, and then gone back and actually done the things that have been suggested, you should have Eclipse up and running, and you should have a working project, and you should be able to clone that project so you can mess around with the new one without worrying about what happens to it.

You should understand by now a little bit about the way the project fits together. You know about the RootPanel, you know how to add things to slots.

You have got something to play with. That's not bad for a first day's work.

If you have any problems, then I am at the end of an email. I will answer as soon as I can, but there is only me there, and I need to sleep sometimes.

## **Homework**

There is no homework, of course. This is not a school. If you want to play with GWT, then you can try my site and have a look at some of the widgets that you can use.

<http://examples.roughian.com/#Widgets~Summary>

## **Next Time**

Tomorrow we'll be playing with simple widgets.